05 月 31 日

第六章：走进自定义样式的世界（下）

| 本文为付费栏目文章，您已订阅，可阅读全文 |

编注：这篇文章作者将会用少数派网站的 CSS 来带大家入门自定义样式，篇幅较长，也有一定的代码含量。如果您是代码小白，不用担心，这部分内容整体难度不高，作者的讲解也很细致，希望您可以踏实地掌握每个细节。计算机上很多地方的自定义现在都用到了 CSS 或者比 CSS 更轻量的方式，学好这篇将让您今后对「自定义」这件事从理解到动手能力都提高不少。

经过上一章的介绍，相比你对 CSS 样式文件有了感性的认识。CSS 绝不是什么普通用户无法企及的高深秘笈，而只是把我们日常习惯用鼠标在界面上进行的操作，化成了可读性更强的文本而已。

但到目前为止，我们还没接触 CSS 的具体写法。在这一章，我们将以**制作出少数派网站的 CSS 样式**为目标，去学习：

1. CSS 中各类选择器用法和区别；
2. 常见的选择器和属性的写法。

正如之前所言，CSS 中可选的选择器和属性多到能让你立刻从入门到放弃。但若掌握了其中的原理，便能起到举一反三的效果。

从模仿开始学习 CSS

虽说我们的目标是「写一份少数派网站的 CSS 文件」，但若真要从头开始，通过观察网页自己去反向推断出网页的 CSS，难度就像对着餐厅里出品的菜，自己反推出菜的制作过程。这么去做不是不可能，但也需要对 CSS 语法有着相当地熟悉。

而如果能有一份「菜谱」，能帮助我们搞清制作流程，无疑会减少很多不必要的步骤。这份「菜谱」就是少数派网站后台使用的 CSS 文件。

而我们在这一章的任务，就是跟着这份菜谱，将它**转换成 Ulysses 能理解的 CSS 文件**。

获得少数派的 CSS 文件

少数派网站的 CSS 样式可以非常方便的获取。

下载少数派的 CSS

利用 Safari （或 Chrome ）打开少数派任意一篇文章网页，点击「开发 – 显示网页检查器」（快捷键 `⌘Command + ⌥Option + I`），在网页检查器中选择「资源 – 样式表」，就能看到当前网页使用的 CSS 文件。可以看到当前有四个 CSS 文件，我们选择下载 `ss_article_style.css` 文件，它将是我们主要参考的文件。

了解 CSS 中灵活多变的选择器

打开文件后你会发现，`ss_article_style.css` 中选择器的名称和之前介绍的有很大不一样，而且每一种选择器前都有`.wangEditor-txt` 字样。而且之前说我们的任务是「把网站的 CSS 转换成 Ulysses 能理解的 CSS 」，如果都是 CSS 格式文件，难道不能直接在 Ulysses 中使用吗？

直接使用少数派 CSS 文件的预览效果

是骡子是马，拉出来遛遛一试便知。把 `ss_article_style.css` 文件导入 Ulysses，预览一篇文稿（预览快捷键是 `⌘Command + ⇧Shift + P`），会发现渲染出来的样式和少数派网站上完全不同。

在上一章中我们见过了 Ulysses `Helvetica Neue` 样式文件中对 `body` 的介绍，而在 `ss_article_style.css` 中的第一部分则是：

```
.wangEditor-txt {
  width: 100%;
  text-align: left;
  padding: 0;
  margin-top: 5px;
  font-family: -apple-system, BlinkMacSystemFont, PingFang-SC-Regular, '
Hiragino Sans GB', 'Microsoft Yahei', Arial, sans-serif;
  font-size: 16px;
  color: #4a4a4a;
  line-height: 26px;
  word-break: normal;
  word-wrap: break-word;
}
```

仔细观察，这部分内容中包含了字体（font–family)、字号（font – size）、颜色（color）等属性。其实这部分内容就是用来**设定样式的主体内容**的，和之前介绍的选择器 `body` 的作用一致。

但为什么在 `ss_article_style.css` 中它要叫做 `.wangEditor-txt` 呢?

这种以点号 `.` 开头的选择器，被称为「类选择器」（.class Selector）。我们先不去深究它的定义，从实际效果来看，开发者通过**自行创建**一个选择器，**取代** 了原本 `body` 的作用，就好像给官方规定的元素选择器，取了一个别名。

再往下看 `ss_article_style.css` 文件，会发现很多熟悉的元素选择器，例如标题（h1 – h6）、图片（img）等，他们都被加上了 `.wangEditor-txt` 作为前缀。从最终效果来看，`ss_article_style.css` 中的 `.wangEditor-txt img` 和 `Helvetica Neue` 样式中的 `img` 部分完全一致，只是名字不同。

之所以使用类选择器，而不是 W3C （World Wide Web Consortium，万维网联盟，制定 CSS 标准的机构）给出的选择器，是因为开发者在开发网站的

过程中，经常会需要创建一些特有的样式（不在标准给出的范围内的样式）。使用类选择器，能让 CSS 文档中的命名规则保持一致，对于大项的网页开发是很有实际意义的。

而 Ulysses 对 CSS 的解析，是建立在 W3C 官方标准上的。少数派的网站后台可以解析自己的 CSS，而**到了 Ulysses 中，就没法知道少数派开发者创建的类选择器想表达什么内容**，所以在 Ulysses 中，没法直接使用 `ss_article_style.css` 文件。

虽然因为选择器名称不同，导致没法直接使用网站上下载的 CSS 文件，不过这正好也给我们指明了努力的方向：将 `ss_article_style.css` 转换成 Ulysses 能理解的 CSS 格式即可。

动手制作一份 CSS！

一份 CSS 通常长达数百行，但都是**由一个个选择器**组成。下面我们便开始采取个个击破的策略，制作出少数派网页风格的 CSS 文件。

打开 TextMate，新建一份 CSS 文件，命名为 `sspai Style`，就开始动手了！

阅读建议：下文的各小节我将先以 `ss_article_style.css` 中的选择器为例，讲解它们的含义和用法。然后再给出 `sspai Style` 中该写的内容。各小节之间没有严格的前后顺序，各位可以根据自己的习惯阅读。

⬇️ **点此下载最终制作成的样式文件**

从最特殊的 Body 开始

对于一份文章样式，首先要设定的是 `body` 部分，也就是文章的主体内容。

`ss_article_style.css` 中的设定

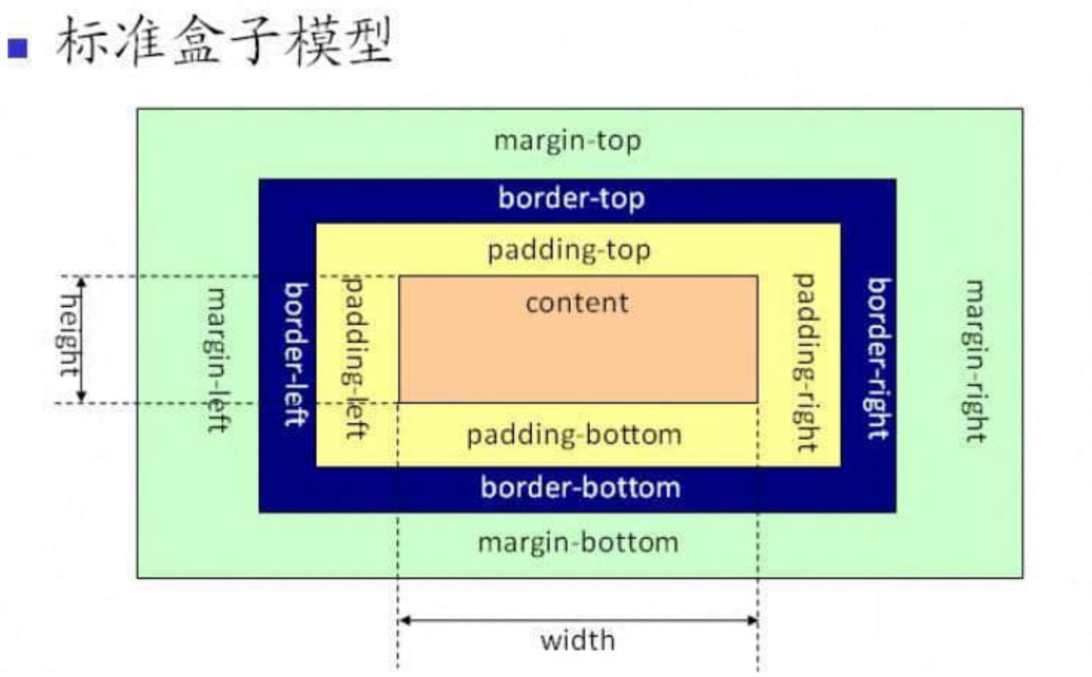
```
  .wangEditor-txt {
  width: 100%;
  text-align: left;
  padding: 0;
  margin-top: 5px;
  /*font-family: PingFang SC, Helvetica Neue, Helvetica, Hiragino Sans GB, Microsoft YaHei, Arial, sans-serif;*/
  font-family: -apple-system, BlinkMacSystemFont, PingFang-SC-Regular, 'Hiragino Sans GB', 'Microsoft Yahei', Arial, sans-serif;
  font-size: 16px;
  color: #4a4a4a;
  /*font-weight: 400;*/
  line-height: 26px;

  word-break: normal;
  word-wrap: break-word;
```

```
}
```

先来逐个分析出现的属性与相应的值。

- `width`：设置页面的宽度。使用百分比是值的常见写法之一，`100%` 意味文章则意味着和网页同宽。一般网页的宽度为 768px（px：像素，图像显示的单位），所以可以将宽度设定为 `width: 768px`，这样可以在 Ulysses 中直接模拟网页上的效果。
- `text-align`：文本的水平对齐方式，`left` 意味着左对齐。在 CSS 网页文档中，可以查看其它值。
- `padding`：内边距。如下图所示的盒子模型，包围在网页内容的一环就是内边距的范围。而 `ss_article_style.css` 中的内边距为零。


标准盒子模型
margin-top
border-top
padding-top
content
height
margin-left
border-left
padding-left
padding-right
border-right
margin-right
padding-bottom
border-bottom
margin-bottom
width

CSS 的盒子模型

- `margin-top`：上外边距。在上图可以清楚的看到上外边距所处的范围。除了用百分比表示，网页开发时使用像素来作为单位也是非常常见的。少数派的 CSS 就设定上外边距为 5 个像素点。
- `font-family`：之前也介绍过的文章字体。这里设置多个字体，是防止设定的字体某些浏览器不支持。例如 Windows 系统就无法支持「PingFang-SC-Regular」。值得注意的是，如果值是由多个单词组成，就必须用双引号包裹起来，否则会无法正确识别。
- `font-size` 和 `line-height`：字号和行高。使用了像素来作为单位设定。

- `color`：字体颜色。在 CSS 参考文档中，介绍了多种表示 CSS 颜色的方式。其中最常用的还是十六进制颜色值。CSS 颜色规范中定义了 147 中颜色名，可以在文档中查看。另外介绍一个小技巧，macOS 系统中自带「数码测色计」应用，可以直接获取屏幕上某块颜色的 RGB 值。将 RGB 作为 `color` 属性的值同样可行。
- `word-break`：自动换行的处理方法。`normal` 表示使用浏览器默认的换行规则。可以在 CSS 参考文档中查看其它两种方式。
- `word-wrap`：长单词或 URL 地址换行的处理办法。`break-word` 表示在长单词或 URL 地址内部进行换行。

`sspai Style` 中的设定

分析完 `body` 内的属性，发现它并没有针对段落、图片或者脚注等设定样式，在 Markdown 中，也没有 `body` 对应的的语法标记，那么它的作用是什么呢?

`body` 的第一个作用，是**对文档布局的设定**。例如宽度（width）、对齐方式（text-align）和边距（margin）等属性，直接决定了渲染后的文章版式。

`body` 的第二个作用，则是**作为其它文章元素的基本样式**。例如字体（font-family）、字号（font-size）等属性，如果 CSS 中没有对段落或者标题等要素的字体字号进行特别设置，Ulysses 就会使用 `body` 中的设定。

我们直接将 `ss_article_style.css` 中 `.wangEditor-txt` 改为 `body`，因为不需要考虑多浏览器和平台的适配，对 font-family 属性的值稍加修改，即可完成这部分的设定。

```
body {
  width: 768px;
  text-align: left;
  padding: 0;
  margin-top: 5px;
  margin-left: auto;
  margin-right: auto;
  font-family: PingFang-SC-light, 'Hiragino Sans GB', 'Microsoft Yahei',
 Arial, sans-serif;
  font-weight: lighter;
  font-size: 16px;
  color: #4a4a4a;
  line-height: 26px;
  word-break: normal;
  word-wrap: break-word;
}
```

学会使用 CSS 参考手册

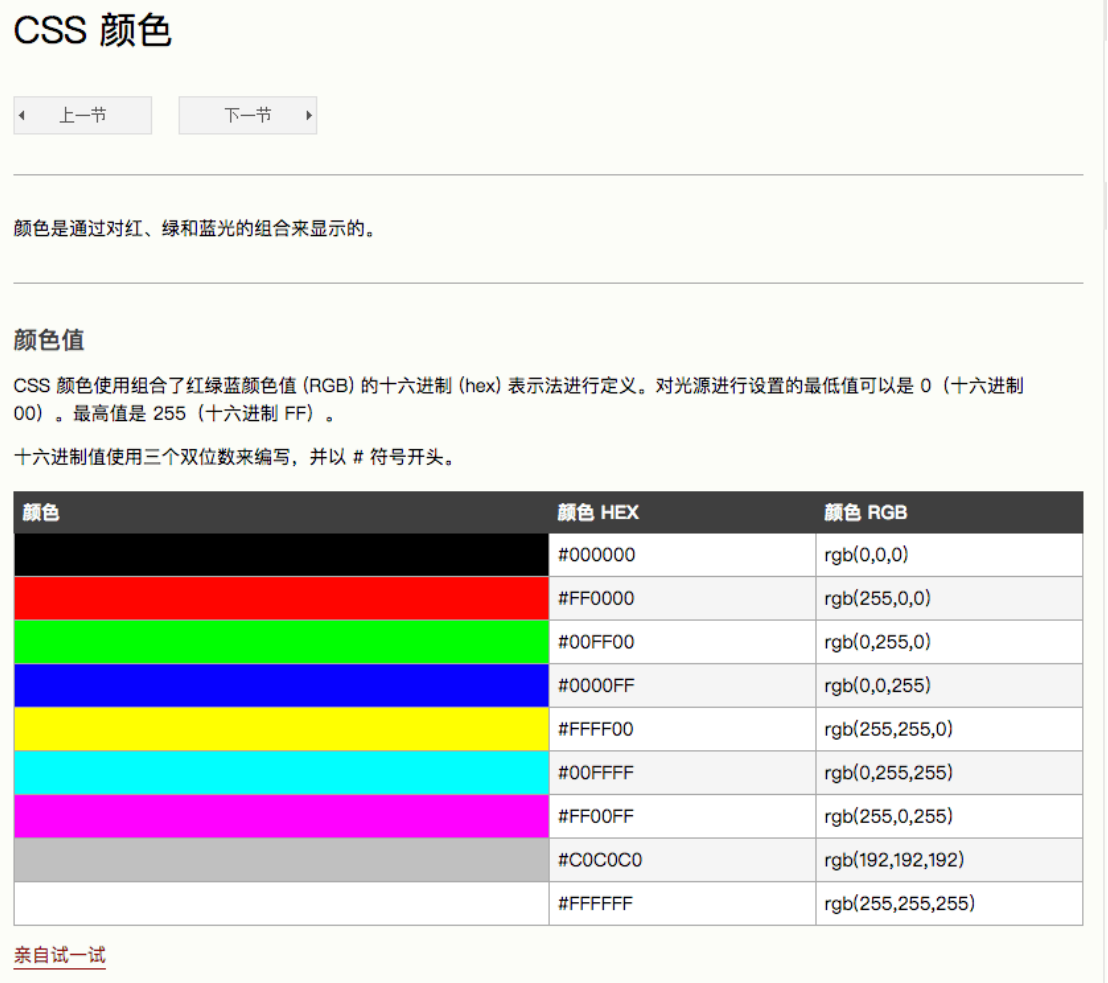
将 `ss_article_style.css` 中对 `body` 的设定，和之前的 `Helvetica Neue.css` 相比，会发现同样是 `body`，但是其中涉及的属性却有不同。而且上一章不是说 CSS 中有上百种属性吗，为什么只设定了这大约十种?

之前一致强调 HTML 的样式是由 CSS 来控制，但其实作为接受 HTML 和 CSS 的工具，浏览器内部也像 Ulysses 一样，内置了默认样式。有些属性没有写在 CSS 中时，例如上面的例子中只设定了 `margin-top`，`margin` 的其它属性却没有提及，到时候浏览器就会使用默认的样式设置，完成渲染的工作。

同样基于浏览器的默认设置，我们在编写 CSS 时，并不用将一类选择器下所有属性都设定一遍，只需要调整自己想特别定制的属性即可。这就使得 `ss_article_style.css` 和 `Helvetica Neue.css` 对 body 的属性设置有所不同。

不过这种灵活性就对刚接触 CSS 的人造成了一些障碍：到底有多少种属性?每种属性有哪些值呢?

在这份 CSS 参考手册中，给出了所有 CSS 中能使用的属性。我们以字体颜色 color 为例，在页面中参考手册给出了它的定义和用法，最关键的是还有颜色属性能指定的值。



CSS 颜色

颜色是通过对红、绿和蓝光的组合来显示的。

颜色值

CSS 颜色使用组合了红绿蓝颜色值 (RGB) 的十六进制 (hex) 表示法进行定义。对光源进行设置的最低值可以是 0（十六进制 00）。最高值是 255（十六进制 FF）。

十六进制值使用三个双位数来编写，并以 # 符号开头。

颜色	颜色 HEX	颜色 RGB
⬛	#000000	rgb(0,0,0)
🟥	#FF0000	rgb(255,0,0)
🟩	#00FF00	rgb(0,255,0)
🟦	#0000FF	rgb(0,0,255)
🟨	#FFFF00	rgb(255,255,0)
🟦	#00FFFF	rgb(0,255,255)
🟪	#FF00FF	rgb(255,0,255)
⬜	#C0C0C0	rgb(192,192,192)
⬜	#FFFFFF	rgb(255,255,255)

亲自试一试

参考手册中对颜色值的介绍

在接下来学习 CSS 过程中，最好随时参照这份参考手册，帮助自己理解属性和值。

对图片的设定

了解到属性和值的众多选择后，下一个问题自然浮现出来：对于一种选择器，我能使用哪些属性进行设定呢?

根据选择器的作用选择属性

遗憾的是，CSS 参考手册中并没有明确为每一种选择器指定属性选择的范围。而对于开发者自己创建的类选择器，更是不可能提前规定出可选的属性。实际上，我们只能**根据选择器在文章中的作用，来针对性的进行设定**。

在 `Helvetica Neue` 中，可以看到两个和图片有关的选择器设定：

```
img {max-width: 100%;}
figure {margin: 1.5em 0; width: 100%; text-align: center; }
```

通过查阅他们在 HTML 参考手册中的介绍，img 是向网页中嵌入一幅图像，而 figure 虽然直译过来也是图像的意思，但是实际上是可以插入任何形式的内容（图片、视频、图表等）。更重要的区别是，**figure 是块级元素，而 img 是内联元素**。

这两类元素的区别在上一章中有过详细介绍，之所以要加以区分，是因为它们**在排版属性上有所不同**。

figure 作为块级元素，会在页面上像盒子一样占据一个独立的空间。所以它可以设定 margin 中上下左右的外边距属性，来控制这个盒子的位置。而 img 作为行内元素，一般会被装在 figure 盒子当中。自然它就不能设定上下外边距（左右外边距依然有效）。

除了不能为 img 设定上下 margin，类似的还有不能为 img 设定字体样式或者字体字号，因为压根没有文字出现。正如前文所说，没有一份标准的规范来指导我们该用哪些属性，所幸我们有 `ss_article_style.css` 来作为参考。

将图片居中和浮动

少数派网站默认会将超出文档宽度的图片压缩到 768 px 的宽度，但是对于不足 768 px 的图片，是会居中处理。而在目前的 CSS 文件中，这些图片会一律是左对齐，如何在 HTML 中实现图片居中的效果呢?

其实方法也非常的简单，利用 img 的 margin 属性，将两边边距设为一样，图片就自动是处于居中位置了：

```
img {
  max-width: 50%;
  margin-left: auto;
  margin-right: auto;
  display: block;
}
```

其中左右边距被设定为 `auto`，这就是使得不管图片的宽度如何，它始终能自动调节合适的左右边距，使得图片呈现居中效果。而且为了确保显示效果正常，还将 `img` 设定为块级元素（本来是行内元素）。

不过还有更加简单的方法。虽然 CSS 中没有针对图片对齐方式的属性，但是利用 `text-align` 属性，在 figure 中 就可以实现图片居中的效果：

```
figure {
  margin: 0 0 30px 0;
  text-align: center;
}
```

实现图片居中后，当多张图片宽度合在一起没有超过文档宽度时，就能实现简单的图片并排的效果。

除了图片居中，文本环绕在图片尽管在网络写作环境中不常见，但是也还是可以通过 img 的属性实现的。在选择器中加入声明 ：

```
float: right;
```

就能将图片变成浮动体，实现文本环绕的效果。关于这个属性的具体解释，可以常见参考手册。

`img` 和 `figure` 的设定

接下来我们尝试对少数派的 CSS 样式中的 `img` 和 `figure` 这两个元素选择器进行设定。

其实少数派网站上用了比较复杂的方式实现多图并排的效果，但需要脚本和浏览器的支持，这在 Ulysses 中是很难实现的。所以我根据 `ss_article_style.css` 和 `ss_extend_style.css` 中的设定进行了简化，实现了单张图片和少数派网页相同的效果：

```
img {
 max-width: 100%;
 margin-left: auto;
 margin-right: auto;
 display: block;
}

figure {
  margin: 0 0 30px 0;
  text-align: center;
}

figcaption {
  outline: none;
  font-size: 14px;
  color: #999;
  letter-spacing: 0px;
  width: 100%;
  text-align: center;
}

figure img {
  margin-bottom: 0px;
}
```

- `img`：插入图片的选择器。将图片的最大宽度设定为 100%，这意味着最大宽度是与 `body` 一样宽，左右边距设为自动，以实现图片居中的效果。

- `figure`：属性 margin 可以直接设定上下左右的外边距。「0 0 30px 0」依次控制的是上、右、下、左外边距。更详细的解释可以在 CSS 参考手册中查看。

- `figcaption`：控制图片下方的说明文字的样式。大多数之前有过解释，新出现的两个属性是 outline 和 letter-spacing。前者用来控制说明文字的轮廓线，none 表示不需要轮廓线；后者知识控制字符间的空白。

- `figure img`：控制 img 元素在 figure 中的样式。这是一种新出现的用法，下文将会专门解释。

至此**完成了 CSS 中对插入的图片的样式设定**。

对标题的设定

接下来我们对文章的标题进行设定。

将选择器分组

观察 `ss_article_style.css` 文件，发现与 `img` 的写法略有不同，对于从一级标题到六级标题，除了单独对每一级标题的设定，例如：

```
.wangEditor-txt h1 {
  font-size: 30px;
}
```

还有将它们合在一起进行设定：

```
.wangEditor-txt h1,
.wangEditor-txt h2,
.wangEditor-txt h3,
.wangEditor-txt h4,
.wangEditor-txt h5,
.wangEditor-txt h6 {
  margin: 30px 0 10px 0;
  color: #4a4a4a;
  line-height: 1.4;
}
```

这种写法被称为**选择器分组**。

对于第一到第六级标题，虽然字体大小不同，但是其他属性却是相同的：

- 外边距：上下外边距各为 30 像素，和段落保持距离。
- 颜色：查看 CSS 颜色表，`#4a4a4a` 是灰色的一种。
- 行间距：1.4 意味着行间距为当前字体尺寸的 1.4 倍。

对于这些属性，如果分开写在每一级标题中，显得太过繁琐。所以将这些选择器写在一起，之间用逗号 `,` 隔开，就意味着**这些属性设定，同时应用于这组包含的元素选择器中**。

对 h1 - h6 的设定

而对于各级标题中不相同的属性，则再单独设定字体大小。根据 `ss_article_style.css` 中对各级标题的设置，我们可以在 sspai Style 中这样设定：

```
h1, h2, h3, h4, h5, h6 {
  margin: 30px 0 10px 0;
  color: #4a4a4a;
  line-height: 1.4;
}

h1 {
  font-size: 30px;
}

h2 {
  font-size: 24px;
}

h3 {
  font-size: 20px;
}

h4 {
  font-size: 16px;
}

h5 {
  font-size: 14px;
}

h6 {
  font-size: 12px;
}
```

至此**完成了对标题的样式设定**。

其实在 `ss_article_style.css` 中，还有很多采用了选择器分组的方式来设定属性的地方，这里一并将它们解决！

```
.wangEditor-txt b,
.wangEditor-txt strong {
        font-weight: 700;
}
```

- `b` 和 `strong` 元素在 HTML 中都是可以用来通过**加粗字体**来强调文本。在 `body` 中字重被设定为 `lighter`，而为了凸显粗体，在这里将它设定为 700 （字重 normal = 400）。

```
.wangEditor-txt em,
.wangEditor-txt i {
  font-style: italic;
}
```

- `em` 和 `i` 元素在 HTML 中通常用来设定**斜体字体**。`body` 中没有特别设定，我们将 `font-style` 改为「italic」即可。`italic` 即意大利体，每种字体中都会有 italic 来显示斜体文字。

```
  .wangEditor-txt del,
  .wangEditor-txt strike {
          text-decoration: line-through;
  }
```

- `del` 和 `strike` 元素则是用于定义**加删除线文本**。实现方式也非常简单，`text-decoration` 属性可以给文本添加修饰，`line-through` 则是给加一条穿过文本的线。

将**标题、粗体、斜体和删除线**的部分稍加整理，可以在 sspai.com Style 加入相应的内容：

```
h1, h2, h3, h4, h5, h6 {
  margin: 30px 0 10px 0;
  color: #4a4a4a;
  line-height: 1.4;
}

h1 {
  font-size: 30px;
}

h2 {
  font-size: 24px;
}

h3 {
  font-size: 20px;
}

h4 {
  font-size: 16px;
}

h5 {
  font-size: 14px;
}

h6 {
  font-size: 12px;
}

b, strong {
  font-weight: 700;
}
```

```
em, i {
  font-style: italic;
}

del, strike {
  text-decoration: line-through;
}
```

对代码的设定

接下来我们考虑对文章中的代码进行设定。

把行内元素装进「盒子」

在 HTML 中，元素 `code` 用来定义文章中的代码（行内代码）。而

在 `ss_article_style.css` 中，也有着相应的设定：

```
.wangEditor-txt code {
  background-color: #f6f8fa;;
  font-family: Source Code Pro, Consolas, "Liberation Mono", Menlo, Cour
ier, monospace;
  font-size: 14px;
  display: inline-block;
  padding: 0 5px;
}
```

字体（font-family）、字号（font-size）和内边距（padding）等属性之前都

出现过，这里不再赘述。在少数派网页上可以看到，文中的代码都会带有红色

的背景颜色，为了实现这个效果，在 CSS 中将代码的显示方式（display）设定为 `inline-block`。即行内块元素，相当于给代码加上了边框，再设定好背景颜色（background-color），就实现了网页上的效果。

不过 code 是行内元素，只能设定一行文字中出现的代码。如果想表现代码块，该怎么设置 CSS 呢?

其实类似的问题我们之前也遇到过。在插入图片的时候，元素 `img` 是行内元素，被放进了块状元素 `figure` 中，借此在页面中有了单独一块地方用来显示图片。

而类似的，HTML 中通常用 `pre` 元素作为装代码块的「盒子」。被包围在 pre 元素中的文本通常会保留空格和换行符。而文本也会呈现为等宽字体，非常适合用来呈现代码。

使用后代选择器

在 `ss_article_style.css` 文件中：

```
.wangEditor-txt pre code {
  font-size: 14px;
  border: none;
  background: none;
  display: block;
  border-radius: 0;
  font-family: Source Code Pro, Consolas, "Liberation Mono", Menlo, Cour
ier, monospace;
}
```

这一块就是用来设定在 `pre` 中的代码的样式。

非常特别的是选择器的名称，出现了新的写法 `pre code`，和之前出现过的 `figure img` 格式相同。

`pre code` 选择器不会对普通的代码产生效果，而只对 `pre` 标签内包裹的代码产生作用。就好像在说：如果这段 code 属于 `pre` 的一部分，那么它就采用以下属性设定。这就是所谓的「**后代选择器**」（descendant selector）。

用一段简单的 HTML 文本，就能很好的帮助我们理解所谓的后代选择器:

```
<pre>
        <code> i += 1</code>
</pre>

<code> i += 1</code>
```

可以看到，前一行代码 `i += 1` 所属的位置处于 `pre` 的包裹之中，可以理解为 `pre` 的下一级，它将采用 `pre code` 选择器的属性设定。而后一行代码位于 `pre` 之外，就正常使用元素选择器 `code` 的设定。

在后代选择器 `pre code` 中，只对 code 文本进行了设定，而 `pre` 元素本身也需要设置样式：

```
.wangEditor-txt pre {
  background-color: #f6f8fa;
  padding: 11px 20px 13px;
  white-space: pre-wrap;
  word-break: break-all;
}

.wangEditor-txt pre p {
  margin: 0;
  line-height: 22px;
}
```

在选择器 `pre` 中，只有 `white-space` 是新出现的属性，是用来定义空白符，`pre-wrap` 表示保留空白符序列，但是正常地进行换行。因为代码中经常要**用空格来控制缩进**，所以需要对空白元素小心处理。

实现语法高亮

}

虽然少数派网站上并不支持代码块的语法高亮，但在 Ulysses 13 中，已经实现了编辑器内的语法高亮，预览时代码块只有黑白，也不利于阅读。那么，该如何在 CSS 中实现语法高亮呢?

随着 Ulysses 13 的更新，其内置的样式也更新了语法高亮的支持。打开 Helvetica Neue 样式，会看到如下代码：

```
.syntax-all {
}

.syntax-entity {
    color: #6f42c1;
}

.syntax-tag {
    color: #22863a;
}

.syntax-keyword {
    color: #d73a49;
}

.syntax-parameter {
    color: #24292e;
}

.syntax-string {
    color: #003878;
}

.syntax-constant {
    color: #005cc5;
}

.syntax-variable {
    color: #e36209;
}

.syntax-escape {
    font-weight: bold;
    color:      #22863a;
}

.syntax-comment {
    color: #6a737d;
}

.syntax-error {
    color: #b31d28;
}
```

这些选择器恰好和 `ss_article_style.css` 中一样，Ulysses 自定义出来的类选择器。通过名称可以判断，这些选择器是为了给代码块实现语法高亮。例如 `.syntax-variable` 是将变量设为红色。除了这里出现的选择器，在官方文档可以查看所有可以设定的代码类型。

`pre` 和 `li` 的设定

而选择器 `pre p` 也是一个后代选择器，它控制的是在 `pre` 标签中的 `p` 元素。

`p` 元素在 HTML 表示段落，而 `pre p` 自然表示的是在代码块中的文本段落，

这是为了代码块中可能会出现的标注设定的样式。

`ss_article_style.css` 中的后代选择器还有有序列表 `ol li` 和无序列表 `ul li`，用法和属性设置与之前出现过的相似，这里不再赘述。

将代码以及列表的 CSS 稍加整理：

```
pre {
  background-color: #f6f8fa;
  padding: 11px 20px 13px;
  white-space: pre-wrap;
  word-break: break-all;
}

pre p {
  margin: 0;
  line-height: 22px;
}

pre code {
  font-size: 14px;
  border: none;
  background: none;
  display: block;
  border-radius: 0;
  font-family: Source Code Pro, Consolas, "Liberation Mono", Menlo, Cour
ier, monospace;
}

ol, ul {
  padding-left: 2em;
}

ol li, ul li {
  margin: 10px 0;
}
```

至此对**行内代码、代码块和列表**的设定完毕。

对链接的设定

在文本中插入链接，几乎是每篇文章必不可少的要素。而对于带有链接的文本，样式自然也需要特别设定来，以与普通文本显示出差别。

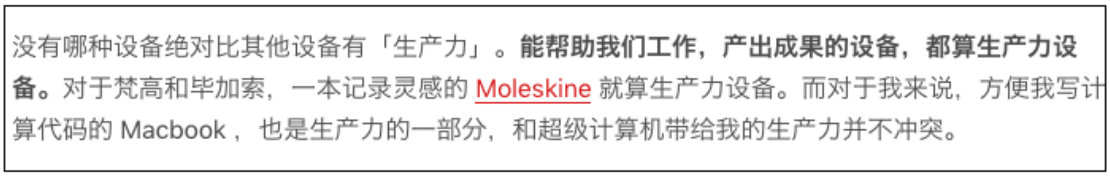
在 HTML 中，`a` 元素通常用来给文体加上链接。自然

在 `ss_article_style.css` 中，就有针对 `a` 元素的选择器：

```
.wangEditor-txt a {
  text-decoration: none;
  border-bottom: 1px solid #4A4A4A;
  color: #292525;
}
```

`border-bottom` 属性是给文本加上了下边框（显示出来就是下划线），对下边框的宽度（1px），线型（solid）和颜色（`#4A4A4A`）都进行了设定。

不过在少数派网页上，但光标移到带链接的文本上时，会发现文本和下边框都会变成红色，这个效果又是如何实现的呢？



没有哪种设备绝对比其他设备有「生产力」。**能帮助我们工作，产出成果的设备，都算生产力设备。**对于梵高和毕加索，一本记录灵感的 Moleskine 就算生产力设备。而对于我来说，方便我写计算代码的 Macbook， 也是生产力的一部分，和超级计算机带给我的生产力并不冲突。

少数派上的链接文本效果

用伪类增强你的选择器

在 `ss_article_style.css` 中，使用的是伪类（Pseudo-classes）来实现。名字听上去奇怪，但可以简单理解为**给某些选择器添加特殊的效果**。

在 `ss_article_style.css` 中就使用了 `:hover` 伪类：

```
.wangEditor-txt a:hover {
  color: #D7191A;
  border-bottom-color: #D7191A;
}
```

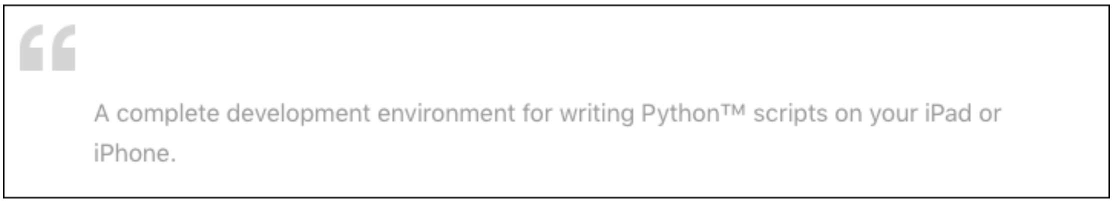
`:hover` 前面加上 `a`，表示当鼠标移到 `a` 元素上时，向此 `a` 元素添加特殊的样式。其中的属性设定就是给字体和下划线设定红色（`#D7191A`）。整段 CSS 代码翻译过来就是：当鼠标移到带链接的文本上时，字体和下划线就变成红色。

`ss_article_style.css` 中使用的伪类不止 `:hover` 一种。对于引用，首先设定了引用文本以及引用中的带链接文本（使用了后代选择器）的样式：

```
.wangEditor-txt blockquote {
  position: relative;
  display: block;
  /*padding: 64px 45px 35px 48px !important;*/
  padding: 44px 45px 25px 48px !important;
  color: #9B9B9B;
  /*margin: 20px 0;*/
  margin: 20px 0 0 0;
  font-size: 16px;
  line-height: 26px;
  border-left: none;
}

.wangEditor-txt blockquote a {
  color: #9b9b9b;
  border-bottom: 1px #9b9b9b solid;
}
```

在少数派网站上，每段引用前面都会有个大引号：



A complete development environment for writing Python™ scripts on your iPad or iPhone.

少数派中引用的效果

这个效果其实就是用 `:before` 伪类实现的：

```
blockquote:before {
  position: absolute;
  top: 0;
  left: 0;
  content: '';
  background: url("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAGAAAAB
QCAYAAADm4nCVAAAAAXNSR0IArs4c6QAABPhJREFUeAHtm4du4zAMhhU33Xuj6Hr/B+oDdO
+95919AmykvTRmYtI2DBII7F4Yivx/iqJkX+vPPwkVydvbW3h5eQlcX19fw/v7e/j8/AxfX
1/xs7CwEJaXlyvyrviwxEF8xJbG+PHxkcVHrO3iw8gtMPjDw0N4fHyMH/7uJRXmRi+3en6X
xsYV8PNiMCcAB+7v78PNzU0EPc+hntHV9EsynPhub29DXlL9DMGMAKbX5eVluL6+jmXl58B
N+JvEuri4CM/PzwOHo04AGZACTw1sotzd3UXgKTFFRZWAq6urcH5+3tiMp9QcHx+Hp6enor
hnv1chgCmIYxoZkXlWoxvWLRKLma29hhUmAKfOzs7UHasL/rSPBwcHZsk1MAHU98PDw9jh1
AUsbT+o9UdHR7Fv17ad2huIADZMu7u7cXORGmralVlNh2MtfRPAQgT4/fa71oFo2mc9o30u
Q/oigNV/f3+/sV0OgFPvKT1lSSIdiMxvOvisaWWCD/YiAqj5e3t7jc7809PTeJQgTUgtvVw
C6HYAHxKaKmwgaaerkFwCaMMoP00VNpFkf1XSkwA6gbJrYplApHsZ7d1tPzH82gWR9WVlRq
vVChMTE2FsbCwMDw+HdrsdP6Ojo/3E0rcus5udbhlCXJOTkzG+zhh/JYBe2PI0M0mS6ND09
HTgMzQ0VAYO2Rg8GLKe3SRQGt/4+Hg2dudNVwJ4sKB54tc5INk+OzsbVlZWYpZ3flfWPSXn
5OTEbLiRkZGwuroawc8b5D8CyHqr0jM1NRUdsy4teUHT8ViUHkonz7Dn5+fzXMi+/48AnNM
+ZiDr19bWwtzcXDZwVTc8qbM446G+b2xs9F1KvxHA1KQn1hSyAsdYZOsgxKe9tpHxJNgg8o
0A7ee3lJqtra248g/inPZvtBOMmU2t5/WZQeUbAZq7QTJ/e3u7soW2GyDaCUa9LwI+PmYbM
d5j0TpuIDM2NzdrBT7B8uqIlszMzISlpaXC5jICNJ2jHv7W9xb2eEADbCy1nlmzYVxfXx/Q
k+8/iwRQG3nHRUPo8evQ7fyMhb2NhjC7aSq4akgkgF2hRmeAU2yw6ihaCUbHw0ZLSyIB1H8
NwTnOOeom7Gs0TnQ5PtF+WViNAAvntIjUSrDFxcW+N1p5MSTsDDWyg9pf9oFaXnDp9xrnWp
TXoi1n6k/nNdHqDDj1q6sUeXk2jYmdvEWCJRrZnx4tp87W7apx8GaVYImGc5xyarVl2uSxA
Gt0eMRoIWoEWDinYVNjhtN2araenXElGkfPnPvUVWgyiopla51oTE9LB4uCpxGfZYKpEGDp
YFECNGaAZXxxH1AkSBZfi/asiE+dv637DGjt7OxU9v+EO4Hqds9xb9GzpbJeM+/mv+Tf4lG
ERNF1bBBwAmxwFVt1AsRQ2Sg6ATa4iq06AWKobBSdABtcxVadADFUNopOgA2uYqtOgBgqG0
UnwAZXsVUnQAyVjaITYIOr2KoTIIbKRtEJsMFVbNUJEENlo+gE2OAqtuoEiKGyUXQCbHAVW
3UCxFDZKDoBNriKrToBYqhsFJ0AG1zFVp0AMVQ2ik6ADa5iq06AGCobRSfABlexVSdADJWN
ohNgg6vYqhMghspG0QmwwVVs1QkQQ2Wj6ATY4Cq26gSIobJRdAJscBVbdQLEUNkoOgE2uIq
tOgFiqGwUnQAbXMVWnQAxVDaKfwGN0SoDfoFaqQAAAABJRU5ErkJggg==") left top no-
repeat;
  background-size: contain;
  width: 36px;
  height: 40px;
}
```

`blockquote:before` 表明要在引用之前插入新内容。而插入的内容其实就是网页上显示的引号的图片，并将它作为背景。`background: url()` 中的一长串字符，是将引号的图片用 base64 的编码格式来表示。通过 `blockquote:before` 选择器，就实现了和网页上相同的引用效果。

`a` 和 `blockquote` 的设定

对链接文本和引用的 CSS 内容稍加整理：

```
a {
  text-decoration: none;
  border-bottom: 1px solid #4A4A4A;
  color: #292525;
}

a:hover {
  color: #D7191A;
  border-bottom-color: #D7191A;
}

a:focus, a img {
  display: block;
}

blockquote {
  position: relative;
  display: block;
  padding: 44px 45px 25px 48px !important;
  color: #9B9B9B;
  margin: 20px 0 0 0;
  font-size: 16px;
  line-height: 26px;
  border-left: none;
}

blockquote a {
  color: #9b9b9b;
  border-bottom: 1px #9b9b9b solid;
}

blockquote:before {
  position: absolute;
  top: 0;
  left: 0;
  content: '';
  /* 引用的图片 */
  background: url("data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAGAAAAB
QCAYAAADm4nCVAAAAAXNSR0IArs4c6QAABPhJREFUeAHtm4du4zAMhhU33Xuj6Hr/B+oDdO
+95919AmykvTRmYtI2DBII7F4Yivx/iqJkX+vPPwkVydvbW3h5eQlcX19fw/v7e/j8/AxfX
1/xs7CwEJaXlyvyrviwxEF8xJbG+PHxkcVHrO3iw8gtMPjDw0N4fHyMH/7uJRXmRi+3en6X
xsYV8PNiMCcAB+7v78PNzU0EPc+hntHV9EsynPhub29DXlL9DMGMAKbX5eVluL6+jmXl58B
N+JvEuri4CM/PzwOHo04AGZACTw1sotzd3UXgKTFFRZWAq6urcH5+3tiMp9QcHx+Hp6enor
hnv1chgCmIYxoZkXlWoxvWLRKLma29hhUmAKfOzs7UHasL/rSPBwcHZsk1MAHU98PDw9jh1
AUsbT+o9UdHR7Fv17ad2huIADZMu7u7cXORGmralVlNh2MtfRPAQgT4/fa71oFo2mc9o30u
Q/oigNV/f3+/sV0OgFPvKT1lSSIdiMxvOvisaWWCD/YiAqj5e3t7jc7809PTeJQgTUgtvVw
C6HYAHxKaKmwgaaerkFwCaMMoP00VNpFkf1XSkwA6gbJrYplApHsZ7d1tPzH82gWR9WVlRq
vVChMTE2FsbCwMDw+HdrsdP6Ojo/3E0rcus5udbhlCXJOTkzG+zhh/JYBe2PI0M0mS6ND09
HTgMzQ0VAYO2Rg8GLKe3SRQGt/4+Hg2dudNVwJ4sKB54tc5INk+OzsbVlZWYpZ3flfWPSXn
5OTEbLiRkZGwuroawc8b5D8CyHqr0jM1NRUdsy4teUHT8ViUHkonz7Dn5+fzXMi+/48AnNM
+ZiDr19bWwtzcXDZwVTc8qbM446G+b2xs9F1KvxHA1KQn1hSyAsdYZOsgxKe9tpHxJNgg8o
0A7ee3lJqtra248g/inPZvtBOMmU2t5/WZQeUbAZq7QTJ/e3u7soW2GyDaCUa9LwI+PmYbM
d5j0TpuIDM2NzdrBT7B8uqIlszMzISlpaXC5jICNJ2jHv7W9xb2eEADbCy1nlmzYVxfXx/Q
k+8/iwRQG3nHRUPo8evQ7fyMhb2NhjC7aSq4akgkgF2hRmeAU2yw6ihaCUbHw0ZLSyIB1H8
NwTnOOeom7Gs0TnQ5PtF+WViNAAvntIjUSrDFxcW+N1p5MSTsDDWyg9pf9oFaXnDp9xrnWp
TXoi1n6k/nNdHqDDj1q6sUeXk2jYmdvEWCJRrZnx4tp87W7apx8GaVYImGc5xyarVl2uSxA
Gt0eMRoIWoEWDinYVNjhtN2araenXElGkfPnPvUVWgyiopla51oTE9LB4uCpxGfZYKpEGDp
YFECNGaAZXxxH1AkSBZfi/asiE+dv637DGjt7OxU9v+EO4Hqds9xb9GzpbJeM+/mv+Tf4lG
ERNF1bBBwAmxwFVt1AsRQ2Sg6ATa4iq06AWKobBSdABtcxVadADFUNopOgA2uYqtOgBgqG0
UnwAZXsVUnQAyVjaITYIOr2KoTIIbKRtEJsMFVbNUJEENlo+gE2OAqtuoEiKGyUXQCbHAVW
3UCxFDZKDoBNriKrToBYqhsFJ0AG1zFVp0AMVQ2ik6ADa5iq06AGCobRSfABlexVSdADJWN
ohNgg6vYqhMghspG0QmwwVVs1QkQQ2Wj6ATY4Cq26gSIobJRdAJscBVbdQLEUNkoOgE2uIq
tOgFiqGwUnQAbXMVWnQAxVDaKfwGN0SoDfoFaqQAAAABJRU5ErkJggg==") left top no-
repeat;
  background-size: contain;
  width: 36px;
  height: 40px;
}
```

至此对链接文本和引用的设定完毕。

对脚注的设定

在 HTML 中，没有专门针对脚注的元素，而是将它在文中的上标，变成一个带链接的文本，链接指向页面底部的文字，这样就形成了脚注的效果。

所以在 CSS 文件中，为脚注进行样式设定也分为两部分：文章中的上标以及文章末尾的脚注文字。在 `ss_article_style.css` 中，可以直接看到对上标文字的样式设定：

```
.wangEditor-txt sup {
  color: #fff;
  background-color: #4A4A4A;
  border-radius: 2px;
  font-size: 10px;
  line-height: 1;
  padding: 1px 5px;
  top: -0.8em;
  margin-left: 2px;
  margin-right: 2px;
  text-decoration: none;
  cursor: help;
  position: relative;
}
```

选择器 `sup` 中对文本颜色（color）、背景颜色（background-color）或者行高（line-height）等属性进行了设定。其中 `border-radius: 2px` 是给上标加上了一个圆角边框。值得注意的是 `cursor` 属性，它的值 `help` 使得鼠标移到上标上时，光标就会变成一个问号的形状。

这里要说明的是，在少数派网页上，点击上标会直接弹出脚注而不是跳到文章末的脚注，这需要网站后台的脚本支持，因此在 Ulysses 中是很难实现的。我们直接使用 Ulysses 内置支持的点击上标跳到文章末的方式。

使用 ID 选择器

接下来的难题就是对文章末的脚注文字进行样式设定。难点在于，文末的脚注看上去是一份列表，但明显和正文中的列表不同，想使用后代选择器或者自己创建类选择器都比较麻烦。

此时不妨打开 Ulysses 内置的 `Helvetica Neue.css` 文件，看看它是怎么对文末脚注进行样式设定的。其中这两行是关键：

```
ol#footnotes {font-size: 0.75em;padding-top: 1.5em;margin-top: 3em;margin-left: 0;}
ol#footnotes:before {content: "——"; letter-spacing: -4px;margin-left: -1.5em;}
```

这里又出现了新的选择器类型——ID 选择器。想搞明白它的用法就必须回到 HTML 中去。

其实在 Ulysses 中输出 HTML 文档，当你创建脚注时，在文末 Ulysses 会这样将它翻译成 HTML：

```
<ol id="footnotes"><br/>
 <li id="fn1"> 脚注的内容 <a href="#ffn1"></a></li><br/>
</ol>
```

注意看在有序列表的 `ol` 标签中，被加上 `id="footnotes"`，表明这份有序列表，是一份带有 ID 的特殊列表。不再遵循一般的有序列表的设定，而是去专门的 `ol#footnotes` 这个选择器中找到属性设定。正因为用 ID 注明的方式将它从普通列表挑出来，所以这种选择器被称为 ID 选择器。

`#footnotes` 的设定

在 `ss_article_style.css` 中，并没有针对脚注文本的属性设定，而是放在了 `index 222` 样式表中（如果你还记得，少数派网站后台是由四份样式表的）。经过整理，对脚注文本的 CSS 内容是：

```
ol#footnotes {
  font-size: 14px;
  padding-top: 1em;
  margin-top: 1em;
  margin-left: 0;
  border-top: 1px solid #eaeaea;
  counter-reset: footer-counter;
  list-style: none;
  color: #555;
  padding-left: 5%;
  margin: 20px 0;
}
```

```
ol#footnotes li {
  margin-bottom: 10px;
  margin-left: 16px;
  font-size: 14px;
  font-weight: 300;
  line-height: 26px;
  list-style-type: decimal;
}
```

至此针对脚注的设定完毕。

@规则

最后我们来简单了解下 CSS 中的 @规则。

早期网页基本上是为了电脑屏幕为设计的，但是平板、智能手机的发展让屏幕尺寸变得多种多样，网页内容就有必要根据屏幕的尺寸，对网页进行调整。如何触发这个调整呢，就要靠 @规则。

在 `ss_article_style.css` 中，就利用@规则进行了 CSS 的调整：

```
@media screen and (max-width: 767px){
  .wangEditor-txt iframe {
    min-height: 300px;
  }
}
```

`@media screen and (max-width: 767px)` 的含义是：当显示媒介是电脑屏幕，平板电脑，智能手机的屏幕，且最大文档宽度小于767px，这 `iframe` 选择器的最小高度为 300px，而 `iframe` 的默认设定为：

```
.wangEditor-txt iframe {
  width: 100%;
  min-height: 500px;
}
```

不过因为我们的 CSS 要用于 Ulysses 中的预览而非少数派网站，所以建议使用 `Helvetca Neue` 当中的相关设定：

```
@media screen and (max-width: 768px) {
    body {
        margin:30px;
    }
}

@media screen and (max-width: 736px) {
    body {
        margin:20px;
    }
    ul,ol {
        margin: 0;
    }
    h1 {
        font-size: 2em;
    }
    h2 {
        font-size: 1.8em;
    }
}
```

当屏幕的最大宽度小于 738px 和 736px 时，便会触发对 `body`、`ul,ol`、`h1`、`h2` 的修改。

小结

在本章中，我们根据少数派网站的 CSS 文件，从头到尾的将它们转换成了 Ulysses 能理解的文件。

⬇️ **点此直接下载该样式文件**

经过这一章的跋涉，我们总算小有所成。但同时也必须意识到，想熟练掌握 CSS 的使用，还是需要大量时间和经验积累的。

不过有了这一章作为起点，我相信你对 CSS 的构成已经有了理性的认识。之后借助官方参考手册和其它资料，便能开始自己慢慢摸索出自己需要的 CSS 样式。

而有了学习 CSS 的经验，我们便可以放心进入自定义样式的下一阶段 ——Ulysses Style Sheet，看如何用它来定义 PDF 和 DOCX 文档，帮助我们在办公和学生环境中解决排版问题。

我们下一章见！